Hierarchical Strategy of Model Partitioning for VLSI–Design Using an Improved Mixture of Experts Approach

K. Hering & R. Haupt
Universität Leipzig, Inst. für Informatik
Augustusplatz 10/11
04109 Leipzig, Germany

Th. Villmann
Inst. für Techno- und Wirtschaftsmathematik
E.-Schrödinger-Str., Geb. 48/575
67663 Kaiserslautern , Germany

to appear in the Proceedings of the 10th Workshop on Parallel and Distributed Simulation (PADS'96)

KEYWORDS. Partitioning, VLSI–Design, Logic Simulation, Genetic Algorithms, Hierarchical Mixture of Expers Strategy

Abstract

The partitioning of complex processor models on the gate and register-transfer level for parallel functional simulation based on the clock-cycle algorithm is considered. We introduce a hierarchical partitioning scheme combining various partitioning algorithms in the frame of a competing strategy. Melting together different partitioning results within one level using superpositions we crossover to a mixture of experts one. This approach is improved applying genetic algorithms. In addition we present two new partitioning algorithms both of them taking cones as fundamental units for building partitions.

1 Introduction

Logic design for whole microprocessor structures is accompanied with time-extensive simulation processes. Within the design strategy outlined in [14] the verification of functional (logical) behavior is strictly separated from the analysis of timing aspects. In this context the background of the present paper is given by simulation processes for functional design verification on gate and register-transfer level (logic simulation) where sequences of machine instructions or microcode are taken as test cases and underlying models comprise complex parts of processor structures. Under these assumptions the usage of cycle-based simulators is to be preferred. *TEXSIM*[1] is a high performance simulator for logic simulation of synchronous designs using the *clock-cycle algorithm*. To achieve a significant reduction of running time for simulations the task is to parallelize them. Thereby a parallel *TEXSIM* simulation consists of several co-operating *TEXSIM* instances running on loosely coupled RS/6000 processors (system *SP2* of IBM) over parts of the whole model. As a basic assumption, the process of the evaluation of combinational logic during the parallel simulation of a cycle has to be left unchanged. Therefore special *fan-in cones* are chosen as building blocks for model partitioning. A partition is directly related to certain workloads of the processors involved in later parallel simulation and communication overhead between co-operating *TEXSIM* instances and, hence, to the speed-up possible due to parallelization. The amount of time acceptable for partitioning depends on the expected total duration of all simulation runs to be performed regarding to a corresponding model. Simulation processes we are dealing with are characterized by a large number of time-extensive runs concerning a given model.

[1] copyright by IBM

2 Definitions

First, we define a structural model for the logic design on gate and register-transfer level. The underlying hardware is supposed to be synchronous. Basic components are given by the sets M_I, M_O, M_E, M_L, M_S (global inputs, global outputs, logical elements, storing elements, signals). M_E includes all elements which represent combinational logic within the hardware to be simulated. Signals of M_S are interpreted as wires. The elements of the set M_L possess storing function and are cycle limiting in the sense of the *clock-cycle* algorithm. We concentrate the elements of all pairwise disjoint sets M_I, M_O, M_E and M_L to the set of boxes $M_B = M_I \cup M_O \cup M_E \cup M_L$. On the basis of these sets

the hardware model can be considered as a directed bipartite graph. Therefore, we introduce the relation $M_{\mathcal{R}} \subseteq (M_B \times M_S) \cup (M_S \times M_B)$ describing the connections between boxes and signals. Using the sets of successors $\mathcal{N}_G^+(x) = \{y|(x,y) \in \mathcal{R}\}$ and predecessors $\mathcal{N}_G^-(x) = \{y|(y,x) \in \mathcal{R}\}$ for any directed graph $G = (X, \mathcal{R})$ and $x \in X$ we define:

Definition 2.1 *Let M_I, M_O, M_E, M_L, and M_S be pairwise disjoint and nonempty sets. M_B and $M_{\mathcal{R}}$ are defined as above. $M = (M_I, M_O, M_E, M_L, M_S, M_{\mathcal{R}})$ is called **hardware model** if the corresponding directed bipartite graph $G(M) = (M_B, M_S, M_{\mathcal{R}})$ [5, 13] satisfies the following conditions:*

1. $\left\{x|x \in M_B \cup M_S \wedge \mathcal{N}_{G(M)}^-(x) = \emptyset\right\} = M_I$,
2. $\left\{x|x \in M_B \cup M_S \wedge \mathcal{N}_{G(M)}^+(x) = \emptyset\right\} = M_O$,
3. *any directed cycle in $G(M)$ includes at least one element of M_L.*

M_I and M_O are the sets of all sources and sinks of $G(M)$, respectively. Condition 3 ensures the absence of directed cycles only including elements of $M_E \cup M_S$. This corresponds to the exclusion of asynchronous combinational feedbacks.
Due to our parallelization approach, cutting signals of M_S during a partitioning of M is only permitted at cycle-boundaries related to the clock-cycle algorithm. Therefore, we are forced to define basic units for partitioning which are known as *cones* [12, 6, 7] with respect to an arbitrarily chosen hardware model M:

Definition 2.2 *The **fan-in cone** $co_I(x)$ of an element $x \in M_O \cup M_E \cup M_L$ is recursively defined by:*

1. $x \in co_I(x)$,
2. $y \in M_E \wedge \mathcal{N}_{G(M)}^+(\mathcal{N}_{G(M)}^+(y)) \cap co_I(x) \neq \emptyset \rightarrow y \in co_I(x)$.

*The **fan-out cone** $co_O(x)$ of $x \in M_I \cup M_E \cup M_L$ is analogously defined using the sets of predecessors.*

Let us take a *cone* $co(x)$ as a special fan-in cone the head element x of which satisfies $x \in M_O \cup M_L$. All the cones form the set $Co(M)$ as the set of basic units for the partitioning of M. An example illustrating the introduced sets M_I, M_O, M_E, M_L and cones for a simple hardware model is depicted in Fig. 1. The number of cones belonging to $Co(M)$ is $m_c = |Co(M)| = |M_L| + |M_O|$. A box $b \in M_E$ (logical

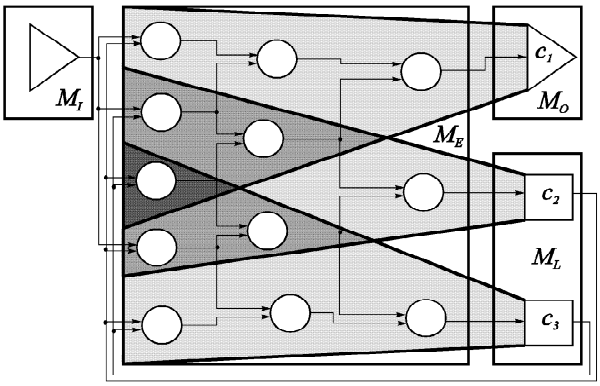


Figure 1 – hardware model with cones (shaded)

element), from which directed paths (with all intermediate boxes being elements of M_E) to the heads of different cones $\hat{c}_i \in Co(M)$, $i = 1, \ldots, m$ exist, belongs to all of the m different cones $\hat{c}_i$: $b \in \cap_{i=1}^m \hat{c}_i$. These cones $\hat{c}_i$ are called to be *overlapping*. Considering all cones c_i of the model we get:

$$\sum_{i=1}^{m_c} |c_i| \geq \left| \bigcup_{i=1}^{m_c} c_i \right| = |Co(M)| + |M_E| \ . \qquad (2.1)$$

If overlapping cones are distributed to different processors one has to take into account the multiple evaluation of boxes in parallel simulations.[2] In the following, $\mathcal{C}$ always denotes nonempty subsets of $Co(M)$.

Definition 2.3 *1.) The **box-related cone overlap degree** $u : M_E \rightarrow \mathbb{N}$ is defined by $u(b) = |\{c|c \in Co(M) \wedge b \in c\}|$, giving the number of cones which contain the box $b \in M_E$.*
*2.) The **overlap region** $ovr(\mathcal{C})$ of a set of cones $\mathcal{C}$ is the set of boxes belonging to these and only these cones $c \in \mathcal{C}$:*

$$ovr(\mathcal{C}) = \left(\bigcap_{c \in \mathcal{C}} c \right) \setminus \left(\bigcup_{c' \in Co(M) \setminus \mathcal{C}} c' \right) \ . \qquad (2.2)$$

All elements of the set $M_E \cup M_L \cup M_O$ are uniquely distributable into overlap regions $ovr(\mathcal{C})$. Using $P^* = 2^{Co(M)} \setminus \{\emptyset\}$ we get:

$$\sum_{\mathcal{C} \in P^*} |ovr(\mathcal{C})| = |Co(M)| + |M_E| \, . \qquad (2.3)$$

[2] On the other hand, communication between the processors is reduced to communication at the clock-cycle boundaries. Of course, replication in the evaluation of boxes is an additional restricting factor and has to be minimized.

In general, most of the overlap regions $ovr(\mathcal{C})$ are empty sets. The set of boxes of a cone c is uniquely decomposable into overlap regions $ovr(\mathcal{C})$ with $c = \bigcup_{(\mathcal{C}\in P^* \wedge c\in\mathcal{C})} ovr(\mathcal{C})$ and $|c| = \sum_{(\mathcal{C}\in P^* \wedge c\in\mathcal{C})} |ovr(\mathcal{C})|$. The set of all overlap regions $ovr(\mathcal{C})$ allows the construction of an equivalent weighted overlap hypergraph GU identifying the nodes with cones and the hyperedges with cone sets $\mathcal{C}$ corresponding to nonempty overlap regions $ovr(\mathcal{C})$ with $|\mathcal{C}| > 1$ [6].
Next, we introduce the terms partitioning and partition by means of two nonempty sets U and V.

Definition 2.4 *1.) A **partitioning** of U with respect to V is a unique map $\Phi : U \to V$ assigning each element $u \in U$ to an element $v \in V$.*
*2.) A **partition** Ψ_Φ of U related to the partitioning $\Phi : U \to V$ is given by $\Psi_\Phi = \left\{\Phi^{-1}(v) \mid v \in \text{cod } \Phi\right\}$, where* cod Φ *is the range of Φ.*

An element $v \in$ cod Φ represents the partition component $\Phi^{-1}(v)$ containing all elements $u \in U$ which are mapped onto v. Here, we identify U with the set of cones $Co(M)$ and V with the set $\mathcal{B}$ of m_b blocks representing processors. If not specified otherwise we consider surjective partitionings.
The task is to find a partitioning $\Phi_{\text{opt}} : Co(M) \to \mathcal{B}$ for the given sets $Co(M)$ and $\mathcal{B}$, which leads to a significant lower running time T_{par} for parallel simulation of a clock-cycle in comparison with T_{seq} in the sequential case. To achieve this goal we consider quality functions taking into account several aspects such as interprocessor communication, workload balance and replication rate influencing T_{par}. Then, for a certain quality function Ω one has to determine a partitioning $\Phi^{\Omega}_{\text{opt}}$ which optimizes Ω. Currently we consider

$$\Omega = \max_{B\in\mathcal{B}} (W(B)) \tag{2.4}$$

as to be minimized where

$$W(B) = \left| \bigcup_{c\in\Phi^{-1}(B)} c \right| . \tag{2.5}$$

$W(B)$ can be interpreted as *workload* of block B under the assumption of an unique time unit $\tau = 1$ for the evaluation of each box. $W(B)$ can be also expressed in terms of overlap regions: $W(B) = \sum_{(\mathcal{C}\in P^* \wedge \mathcal{C}\cap\Phi^{-1}(B)\neq\emptyset)} |ovr(\mathcal{C})|$. For sequential simulation we have with (2.3) the sequential workload as

$$W_{seq} = |Co(M)| + |M_E| \tag{2.6}$$

which is equal to the sum of boxes to be evaluated.

3 Hierarchical Partitioning

3.1 Hierarchical Strategy and the Mixture of Experts Approach

In the applications considered here the ratio between m_c as the number of cones and m_b as the number of blocks may be up to the range of $10^5 - 10^6$. Therefore, we focus onto a hierarchical strategy [1] which has been successfully applied to data extensive problems as, for instance, non-linear principle component analysis (PCA) and robotics [10]. To gradually reduce the range of the problem we introduce a general q–level partitioning scheme according to Def. 2.4:

Definition 3.1 *A **q-level partitioning** of U with respect to V is defined by $\Phi_H : U \to V$ with $\Phi_H = \Phi_q \circ \Phi_{q-1} \circ \ldots \circ \Phi_1$ where $\Phi_j : V_j \to V_{j+1}$ and $V_1 = U$, $V_{q+1} = V$ and furthermore $|V_1| \geq |V_2| \geq \ldots \geq |V_{q+1}|$.*

Clearly, in general Φ_H is only an approximation of Φ_{opt}. In our application we use a 2–level scheme $\Phi_H = g \circ f$, i.e. $V_1 = U = Co(M)$, $V_2 = \mathcal{S}$ and $V_3 = V = \mathcal{B}$:

$$\Phi_H : Co(M) \xrightarrow{f} \mathcal{S} \xrightarrow{g} \mathcal{B} . \tag{3.1}$$

Thereby, $\mathcal{S}$ is a set of elements S_l, the pre–images $s_l = f^{-1}(S_l)$ of which are called *super–cones*. We remark that super–cones are collections of usual cones.[3]
In contrast to the determination of the cones the realizations of g and f are free. This allows an optimal adaptation. However, often an *a priori* optimal choice is not possible [5]. To overcome this difficulty we prefer in each level of the hierarchical scheme a strategy introduced in neurodynamics by JORDAN et al. [2] which is called *mixture of experts*.
For a q–level scheme we consider several partitioning algorithms A^j_i, $i = 1 \ldots m_j$ corresponding to maps Φ^j_i and working in a parallel way in one hierarchical step j representing various partitioning heuristics. The resulting partitions $\Psi_{A^j_i}$ are compared with respect to a quality measure and the β_j best of them will form the basis for the algorithms A^{j+1}_l of the next level which generate partitions $\Psi_{A^j_i, A^{j+1}_l}$. Thereby, the images of the super–cones of a partition $\Psi_{A^j_i}$ given by Φ^j_i are taken as the new basic units. The final result of a q–level scheme is a partition $\Psi_{A^1_{i_1}, A^2_{i_2}, \ldots, A^q_{i_q}}$ the quality

[3] On the other hand, the cones themselves are sets of boxes which are elements of $\mathcal{M} = M_E \cup M_L \cup M_O$. Therefore, we can regard the concentration of these as an initial 'partitioning' $\Phi_0 : \mathcal{M} \to Co(M)$. In this way we obtain $\Phi^*_H = g \circ f \circ \Phi_0$ as an extended 2–level scheme. The definition of the cones uniquely determines the map Φ_0. Yet, Φ_0 is not a partitioning in the sense of Def. 2.4.

measure of which is the best in the last level. However, as yet this describes only a simple strategy of competing experts.
By introducing superpositions $\tilde{\Psi}$ of a set $\Pi = \{\Psi_1, \ldots, \Psi_k\}$ of partitions within a certain level we next extend the competing approach to a mixture one. In this context $\tilde{\Psi}$ plays the role of a generating system, i.e. each super–cone of a partition $\Psi_i \in \Pi$ is expressible in terms of super–cones $\tilde{s}_l \in \tilde{\Psi}$:

Definition 3.2 *Let $\Pi = \{\Psi_1, \ldots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \ldots n_i$. $\tilde{\Psi} = \{\tilde{s}_1, \ldots, \tilde{s}_m\}$ is called a **superposition** of Π if and only if:*

1. *$\tilde{\Psi}$ is a partition of U*
2. *$\tilde{\Psi}$ is a generating system for each $\Psi_i \in \Pi$, i.e., for each $s_i^j \in \Psi_i$ $(i = 1 \ldots k,\ j = 1 \ldots n_i)$ exist $\tilde{s}_{l_1}, \ldots, \tilde{s}_{l_r} \in \tilde{\Psi}$ such that $s_i^j = \tilde{s}_{l_1} \cup \ldots \cup \tilde{s}_{l_r}$.*

Def. 2.4 yields $\emptyset \notin \tilde{\Psi}$. The elements of U taken as single sets form a superposition $\tilde{U}$ of Π. However, we want to have a superposition the granularity of which is rougher than the granularity of $\tilde{U}$, i.e., $|\tilde{U}| > |\tilde{\Psi}|$. Therefore we consider a special construction of superpositions:

Theorem 3.3 *Let $\Pi = \{\Psi_1, \ldots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \ldots n_i$. Furthermore, let Ψ^* be given as*

$$\Psi^* = \left\{ s^*_{j_1 \ldots j_k} \mid s^*_{j_1 \ldots j_k} = \bigcap_{i=1 \ldots k} s_i^{j_i} \right\} \setminus \{\emptyset\} \qquad (3.2)$$

with $j_i = 1 \ldots n_i$. Then Ψ^* is a superposition of Π. Furthermore, for all superpositions $\hat{\Psi}$ of Π with $\hat{\Psi} \neq \Psi^*$ the relation $|\hat{\Psi}| > |\Psi^*|$ is valid, i.e. Ψ^* has the maximum granularity.

Proof: The proof of the theorem is shown in the Appendix.
Following the theorem we are able to determine a superposition Ψ^* of maximum granularity by k–times intersections according to (3.2). Yet, in general we only have $|\tilde{U}| \geq |\Psi^*|$. The structure of Ψ^* depends on the properties of the partitions $\Psi_i \in \Pi$ which represent the different partitioning heuristics (realized by the corresponding algorithms). Hence, all used strategies influence the superposition, i.e., the expert knowledge of the algorithms is mixed in Ψ^*. We add a superposition according to Theorem 3.3 as a special partition to the β_j best of one hierarchical level j so that it may be used in the next level, too.

Returning to our 2–level scheme, the use of a superposition is suitable after the first partitioning level. If we assume that we have various algorithms A_i^1 realizing the maps $f_i : Co(M) \to \mathcal{S}_i$ we obtain $\Psi_i = f_i^{-1}(\mathcal{S}_i)$ according to Def. 2.4. $\mathcal{S}_i$ are sets of the mappings of the super–cones determined by the partitions Ψ_i, respectively. In analogy, we introduce the abstract map $f^* : Co(M) \to \mathcal{S}^*$ where $\mathcal{S}^*$ is representing the set of super–cones $s_l^* \in \Psi^*$ and $\Psi^* = (f^*)^{-1}(\mathcal{S}^*)$. Then each element of the set system $\Sigma_f = \{\mathcal{S}_1, \ldots, \mathcal{S}_{\beta_1}, \mathcal{S}^*\}$ can be taken as a new system of basic units for partitioning in the second level.
However, the above mixture strategy is a very simple one. In the next section we will improve this strategy using genetic algorithms. Thereby, condition 2 of Def. 3.2 becomes important.

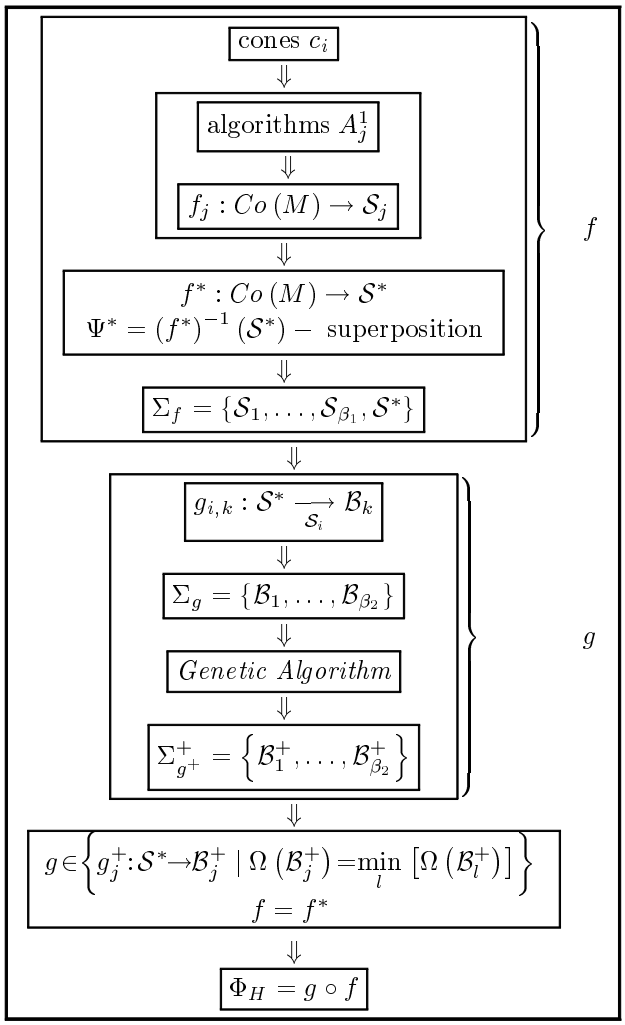
3.2 Improved Mixture of Experts Approach Using Genetic Algorithms

In this part we extend the mixture approach introduced in section 3.1 using *genetic algorithms* (GAs) [1]. In GAs, populations of individuals (parents) produce new individuals (children) in a manner which is inspired by biological evolution and reproduction. The individuals are strings describing a set of parameters which are to be optimized.[4] For applying GAs to graph partitioning let us consider a partitioning map $\Phi : U \to V$. Furthermore, one has to optimize Φ regarding to a certain quality function Ω (fitness function). In this context an individual j represents a certain partition, determined by a map Φ_j. The i-th component of the string is associated with the i–th element of U containing the mapping goal which is an element of V. Several authors have applied GAs to graph partitioning, for instance [8].
However, we will involve this approach into the above described hierarchical strategy. Here we focus onto the 2–level scheme (3.1). In general, GAs may be used in each hierarchical level. Yet, because of the large number of cones in $Co(M)$ the string of an individual representing a partition of $Co(M)$ is often too long for mastering. On the other hand, if applying GAs in the second level of the hierarchical scheme, they require a uniform set of basic elements. To serve this assumption the use of the superposition Ψ^* specified in (3.2) of Theorem 3.3 is appropriate because of its property as a *generating system*. In this context the initial population for the GA is based on the set of all partitions determined in the first level which now are described in terms of the elements of $\mathcal{S}^*$. We emphasize again that the several algorithms represent various

[4] For a more detailed introduction see for instance [4].

partitioning strategies the best of which *a priori* is unknown. Still more, in general a merging strategy will improve the result significantly. We can realize such a strategy using the recombination by *crossing over* in GAs to join different properties of two individuals (partitions) into new ones. The crossing over scheme may be interpreted as a more general exchanging than the simpler one in the algorithm of KERNIGHAN AND LIN [3]. However, we have to take into account a second argument, how much of the old individuals get the chance to be allowed for the competing step (selection) to build the new population. Let us suppose that μ parent individuals produce λ children. Two contrary methods are well known: 1) the μ best of the λ children only form the new population with $\mu < \lambda$; 2) all $\mu + \lambda$ individuals are allowed for the selection process.[5] In the second case the best solution is preserved. Yet, it tends to a stagnation into a local minimum. In the first scheme this property is weakened. On the other hand, good solutions may get lost here. Therefore, we introduce a new so–called $[\mu * \lambda]$–scheme which balances both strategies: at a time t now $\mu_t + \lambda$ individuals have to be taken into consideration with $\mu_t = \text{int}\left[(\mu - \mu_\alpha) \cdot \sigma(t)\right] + \mu_\alpha$. Thereby, $\text{int}[x]$ stands for the integer value of x. The function $\sigma(t)$ is of decreasing sigmoid type with $0 \leq \sigma(t) \leq 1$. μ_α describes the final survival probability for the parent individuals. We have $\mu_0 = \mu$ and $\lim_{t\to\infty} \mu_t = \mu_\alpha$.

The whole procedure, which includes the generation of a superposition and following GA, finally leads to the complete scheme of the improved hierarchical mixture strategy depicted in Tab. 3.2.



Tab. 3.2. : Scheme of the improved mixture of experts strategy using genetic algorithms

3.3 Special Experts

Our mixture of experts approach is a framework for applying several partitioning algorithms as experts. A survey of algorithms suitable for parallel logic simulation is given in [13]. We distinguish *direct* and *iterative* partitioning algorithms which construct a single partition resulting from basic units without building intermediate partitions or require an initial partition which is gradually improved according to a quality function, respectively.

We have developed two new direct algorithms on the basis of cones aiming at balanced workload and minimum replication, the *Backward-Cone-Concentration* algorithm (*n-BCC*) and the *Minimum-Overlap Cone-Cluster* algorithm (*MOCC*).

The basic idea of *n-BCC* consists in iteratively assigning sets of cones to blocks with preferred choice of n cones overlapping each other using the box-related cone overlap degree u of Def. 2.3:

1. Fix a value n^* with smallest distance to n in the range of u; all boxes in M_E are assumed to be unmarked.

2. Choose a box e within all unmarked boxes out of $u^{-1}(n^*)$ and search its *fan-out cone* $co_O(e)$ (see Def. 2.2) to find the head elements of the n^* cones covering e. These n^* cones are assigned to a block possessing the lowest number of cones for the moment and all boxes of the selected cones become marked. If there is a remaining unmarked box $e \in u^{-1}(n^*)$, then step 2 is repeated.

3. If there exists $n' \in \text{cod } u$ with $u^{-1}(n')$ containing an unmarked box, then such a n' is taken as

[5] These correspond to the $[\mu, \lambda]$– and $[\mu + \lambda]$–scheme in the notions of RECHENBERG and SCHWEFEL, respectively [9, 11].

the new n^* and one has to continue with step 2. Otherwise, the algorithm terminates.

Contrary to *MOCC* explained below, *n-BCC* does not explicitly use knowledge concerning the number of boxes in overlap regions or cones. First of all, *n-BCC* has been designed for application at the first level of our hierarchical strategy.
MOCC successively constructs a partition using the specifics of the weighted overlap hypergraph GU corresponding to the hardware model M. With this algorithm the objective is to achieve partitions with blocks containing hypergraph nodes (cones) connected among one another with high-weighted hyperedges:

1. Initially, m_b cones of $Co\,(M)$ are assigned to the m_b blocks.
2. Taking block $B_i \in \mathcal{B}$ with the lowest number of boxes, we are looking for that overlap region $ovr(\mathcal{C}^*)$ of B_i with $\mathcal{C}^* \cap \Phi^{-1}(B_i) \neq \emptyset$ which maximizes the product $|ovr(\mathcal{C}^*)| \cdot \left|\mathcal{C}^* \setminus \bigcup_{j=1\ldots m_b} \Phi^{-1}(B_j)\right|$.
3. Assign the set of these up to now not considered cones $\mathcal{C}^* \setminus \bigcup_{j=1\ldots m_b} \Phi^{-1}(B_j)$ concerning the selected overlap region $ovr(\mathcal{C}^*)$ to the block B_i.
4. If free cones exist yet, proceed with step 2. Otherwise the partition is complete and the algorithm stops.

MOCC aims at a minimum of multiple evaluation of boxes on different processors keeping a balanced workload corresponding to the resulting partition. If two-stage partitioning is necessary the complex structure of GU implies preferably applying *MOCC* to the second level of the hierarchical partitioning scheme.

4 Experimental Results – Conclusions

Finally, we present a special application of the improved mixture of experts strategy (Tab. 3.2) for a specific hardware model M representing a processor structure with $|M_E| = 16\,398$ boxes.[6]
For the initial hierarchical level we use a set of n-BCC algorithms A_k^1 with varying parameters n and numbers of super-cones m_s. The crossing to the second level requires the production of an initial population Σ_g for the genetic algorithm to be applied. Generally, each S_i resulting from the first level allows the production of many individuals $\mathcal{B}_k$ in the second level $g_{i.k} : S^* \underset{S_i}{\longrightarrow} \mathcal{B}_k$, using the elements of S^* as new basic units and keeping such units together in one block of $\mathcal{B}_k$ which correspond to one and the same super-cone belonging to S_i. Here, we restrict the number of created initial individuals $\mathcal{B}_k$ to one for each S_i. For the evaluation of individuals within the genetic algorithm and for choosing the final partition described by Φ_H, the quality function $\Omega \underset{(2.4)}{=} \max_{B\in\mathcal{B}} (W\,(B))$ (maximum workload) is taken.

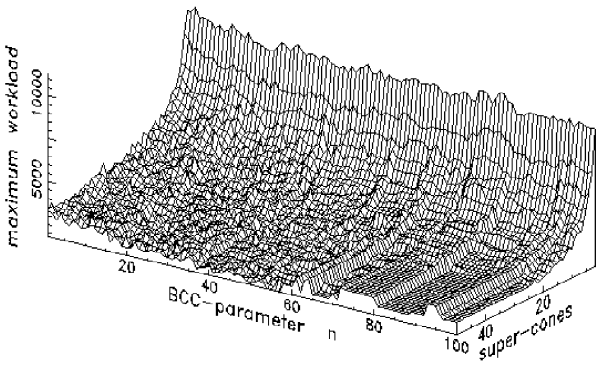


Figure 2 – Quality function Ω (maximum workload) for partitions resulting from the n-BCC for various numbers of super–cones and values of parameters n.

In Fig. 2 the quality function Ω, applied to the partitioning results of the first hierarchical level of n–*BCC* for various $m_s = 2\ldots50$ with the parameters $n = 1\ldots100$, is shown.
In our exemplary application of the hierarchical scheme we investigate 3 cases for application of GAs differing in the initial population which is randomly chosen out of corresponding partitioning results of the first hierarchical level. In all tests the maximum number of blocks is limited to $m_b = 32$.
First, we build the initial population only from partitions consisting of 32 super–cones, i.e. $m_s = 32$. The evaluation in time (number of generations) of the fitness according to the best individual (partition) of the population is shown in Fig. 3 (short dashed line). The Ω–value of the fittest individual decreases from 2250 to 1899 with $m_b = 32$. In the second example all initial partitions are formed over $m_s = 16$ super–cones. Starting with a best individual of $\Omega_{\mathrm{start}} = 2896$ the final solution gives a partition with $\Omega_{\mathrm{end}} = 1985$ and $m_b = 32$ (straight line in Fig. 3). Yet, this is better than the start value in the first case but its final value is not reached. Nevertheless, the difference between Ω_{start} and Ω_{end} in the second example is more than 900 because there is a high variability in crossing

[6] provided by *IBM*

from 16 super–cones up to 32 blocks in a partition. Therefore, in the last experiment we merge individuals of varying parameter n of the n–BCC algorithm on the one hand side and individuals with different numbers m_s on the other hand in the initial population. In fact, this leads to a better performance, i.e. the final value of the fittest individual now is $\Omega_{\text{end}} = 1803$ with $m_b = 32$ again (long dashed line in Fig. 3).

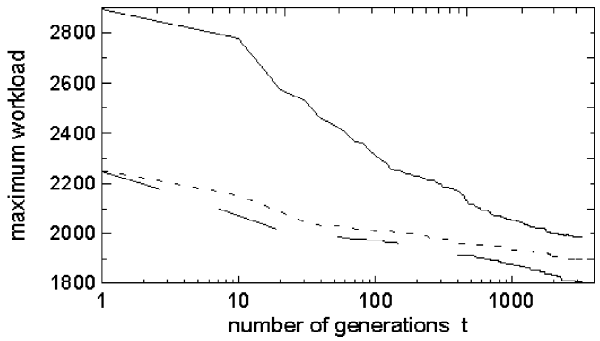


Figure 3 – Maximum workload of the best partition generated by a GA for $m_{s_1} = 16$ (short dashed), $m_{s_2} = 32$ (straight line) and for a mixed initial population (long dashed line) with respect to time (number of generations), see text.

These first results show that a mixture of the *a priori* chosen strategies represented by the various n–BCC instances leads to improved partitions. The successful application of the GAs to the mixture strategy of partitioning algorithms was demonstrated using the idea of superposition of partitions.

Acknowledgement

This work was supported by DEUTSCHE FORSCHUNGSGEMEINSCHAFT (DFG) under the grant **SP** 487/1–2.

5 Appendix

Proof of Theorem 3.3:

(I) Ψ^* is a generating system:

Let $s_i^j \in \Psi_i$ be arbitrarily chosen. We construct the sets S_l with $l \neq i$ according to the following rule: if $s_i^j \cap s_l^{j'} = \tilde{s}_l^{j'}$ with $s_l^{j'} \in \Psi_l$ and $\tilde{s}_l^{j'} \neq \emptyset$ holds then $\tilde{s}_l^{j'} \in S_l$. Then for each l the relation $\bigcup_{j'} \tilde{s}_l^{j'} = s_i^j$ is valid. We consider the set $\Xi^* = \left\{ s^*_{j'_1 \dots j'_k} \mid s^*_{j'_1 \dots j'_k} = \bigcap_{\substack{l=1\dots k \\ l \neq i}} \tilde{s}_l^{j'_l} \; ; \; \tilde{s}_l^{j'_l} \in S_l \right\} \setminus \{\emptyset\}$. Because of the definition of the $\tilde{s}_l^{j'_l}$ as intersections with $s_i^j \in \Psi_i$ we have $s^*_{j'_1 \dots j'_k} \in \Psi^*$ for $s^*_{j'_1 \dots j'_k} \neq \emptyset$ and, furthermore, $\bigcup_{s^*_{j'_1 \dots j'_k} \in \Xi^*} s^*_{j'_1 \dots j'_k} \subseteq s_i^j$. It still remains to show $\bigcup_{s^*_{j'_1 \dots j'_k} \in \Xi^*} s^*_{j'_1 \dots j'_k} \supseteq s_i^j$: We take an arbitrary but fixed $u \in s_i^j$. For each set S_l one and only one $\tilde{s}_l^{j_l^*}$ exists such that $u \in \tilde{s}_l^{j_l^*}$, i.e., $u \in \bigcap_{\substack{l=1\dots k \\ l \neq i}} \tilde{s}_l^{j_l^*}$ with $\bigcap_{\substack{l=1\dots k \\ l \neq i}} \tilde{s}_l^{j_l^*} \in \Xi^*$.

(II) Ψ^* is a partition of U:

Lemma 5.1 *For $s_i^* \in \Psi^*$ and $s_j^* \in \Psi^*$ with $i \neq j$ holds: $s_i^* \cap s_j^* = \emptyset$, i.e. the elements of Ψ^* are pairwise disjoint.*

According to (I) Ψ^* is a generating system for the $s_i^j \in \Psi_i$. Furthermore, it is assumed, that all Ψ_i are partitions of U themselves. Then one can find for each element $u \in U$ a super–cone $s_{j^*}^* \in \Psi^*$ such that $u \in s_{j^*}^*$. Lemma 5.1, the proof of which is shown below, ensures that the elements of Ψ^* are pairwise disjoint.

(III) Ψ^* has the maximum granularity:

Lemma 5.2 *Let Π , $\hat{\Psi}$ and Ψ^* be defined as in Theorem 3.3. Then, for each super–cone $\hat{s} \in \hat{\Psi}$ a super–cone $s^* \in \Psi^*$ exists such that $\hat{s} \subseteq s^*$ is valid.*

For arbitrary elements $u \in U$ and partitions Ψ of U let u^Ψ denote the uniquely determined super–cone $s \in \Psi$ for which $u \in s$ is satisfied. We choose a set $\mathcal{R}$ of elements $u_1, u_2, \dots, u_{|\Psi^*|}$ with $u_j \in U$, $j = 1 \dots |\Psi^*|$ in such a way that for the set $\mathcal{R}^* = \left\{ u_j^{\Psi^*} \mid j = 1 \dots |\Psi^*| \right\}$ the relation $\mathcal{R}^* = \Psi^*$ holds. Using this construction and Lemma 5.2 we have for each of the considered u_j the relation $u_j^{\hat{\Psi}} \subseteq u_j^{\Psi^*}$ with $u_j^{\hat{\Psi}} \in \hat{\Psi}$ and $u_j^{\Psi^*} \in \Psi^*$. The sets $u_j^{\hat{\Psi}}$ form the set $\hat{\mathcal{R}} = \left\{ u_j^{\hat{\Psi}} \mid j = 1 \dots |\Psi^*| \right\}$. Since Ψ^* is a partition it follows that $u_i^{\Psi^*} \cap u_j^{\Psi^*} = \emptyset$ for $i \neq j$ and therefore we get $u_i^{\hat{\Psi}} \cap u_j^{\hat{\Psi}} = \emptyset$. This leads to the inequality $\left|\hat{\Psi}\right| \geq \left|\hat{\mathcal{R}}\right| = |\mathcal{R}^*|$.

Next we derive the corresponding strong inequality. Because of the assumption $\Psi^* \neq \hat{\Psi}$ at least one index j' exists with $u_{j'}^{\hat{\Psi}} \subset u_{j'}^{\Psi^*}$. Consider an element $\hat{u} \in u_{j'}^{\Psi^*} \setminus u_{j'}^{\hat{\Psi}}$. Using Lemma 5.2 and $\hat{u} \in u_{j'}^{\Psi^*}$ we get $\hat{u}^{\hat{\Psi}} \subseteq u_{j'}^{\Psi^*}$.[7] Hence, for each $i = 1 \dots |\Psi^*|$ one has

[7] Moreover, we remark that $u_{j'}^{\hat{\Psi}} \neq \emptyset$ yields because $\hat{\Psi}$ being a partition and, hence, $\hat{u}^{\hat{\Psi}} \subset u_{j'}^{\Psi^*}$ is valid.

$u_i^{\hat{\Psi}} \cap \hat{u}^{\hat{\Psi}} = \emptyset$. This leads to the inequality

$$\left|\hat{\Psi}\right| \geq \left|\overset{\wedge}{\mathcal{R}} \cup \left\{\hat{u}^{\hat{\Psi}}\right\}\right| = \left|\overset{\wedge}{\mathcal{R}}\right| + 1 > |\Psi^*| \qquad (5.1)$$

which completes the proof of Theorem 3.3.□
It remains to show that the Lemmata 5.1 and 5.2 hold:

Proof of Lemma 5.1:

Let $s^*_{i_1\ldots i_k} \in \Psi^*$ and $s^*_{j_1\ldots j_k} \in \Psi^*$ be given with $(i_1,\ldots,i_k) \neq (j_1,\ldots,j_k)$. Then an index l exists such that $i_l \neq j_l$. Further we have $s^*_{i_1\ldots i_k} = s_1^{i_1} \cap \ldots \cap s_l^{i_l} \cap \ldots \cap s_k^{i_k}$ and $s^*_{j_1\ldots j_k} = s_1^{j_1} \cap \ldots \cap s_l^{j_l} \cap \ldots \cap s_k^{j_k}$. Then we obtain for the intersection $s^*_{i_1\ldots i_k} \cap s^*_{j_1\ldots j_k}$ the relation $s^*_{i_1\ldots i_k} \cap s^*_{j_1\ldots j_k} = s_1^{i_1} \cap s_1^{j_1} \cap \ldots \cap s_l^{i_l} \cap s_l^{j_l} \cap \ldots \cap s_k^{j_k} \cap s_k^{j_k}$. The definition of Ψ^* in (3.2) yields $s_l^{i_l} \in \Psi_l$ and $s_l^{j_l} \in \Psi_l$. Because Ψ_l is a partition of U, $s_l^{i_l} \cap s_l^{j_l} = \emptyset$ is valid.□

Proof of Lemma 5.2:

For the proof of the lemma we show that the following assumption leads to a contradiction:

assumption: There exists a super–cone $\hat{s} \in \hat{\Psi}$ such that it could not be found a super–cone $s^* \in \Psi^*$ with $\hat{s} \subseteq s^*$.

Consider a super–cone $\hat{s} \in \hat{\Psi}$ satisfying the above assumption. Because Ψ^* is a partition $|\hat{s}| \geq 2$ follows. In particular, there exist 2 elements u_1, $u_2 \in \hat{s}$ with 1) $u_1 \in s_1^*$ and $s_1^* \in \Psi^*$ and 2) $u_2 \in s_2^*$ and $s_2^* \in \Psi^*$such that $s_1^* \neq s_2^*$ is valid. Let these be given as $s_1^* = \underset{i=1\ldots k}{\cap} s_i^{j_i} = s^*_{j_1\ldots j_k}$ and $s_2^* = \underset{i=1\ldots k}{\cap} s_i^{h_i} = s^*_{h_1\ldots h_k}$. Since $s_1^* \neq s_2^*$ one can find an index $l \in \{1,\ldots,k\}$ for which $j_l \neq h_l$ holds. For the corresponding super–cones $s_l^{j_l}$ and $s_l^{h_l}$ we have: $s_l^{j_l}, s_l^{h_l} \in \Psi_l$ and, hence, we get

$$s_l^{j_l} \cap s_l^{h_l} = \emptyset\,. \qquad (5.2)$$

Furthermore, $\hat{\Psi}$ is a superposition of Π, i.e., one can describe the super–cone $s_l^{j_l}$ in terms of the elements of $\hat{\Psi}$. First, we remark that the relations $u_1 \in \hat{s} \cap s_1^*$, $u_2 \in \hat{s} \cap s_2^*$, $s_1^* \subseteq s_l^{j_l}$ and $s_2^* \subseteq s_l^{h_l}$ lead to

$$\hat{s} \cap s_l^{j_l} \neq \emptyset \quad \text{and} \quad \hat{s} \cap s_l^{h_l} \neq \emptyset \quad . \qquad (5.3)$$

Hence, there exists a decomposition of the super–cone $s_l^{j_l}$ into super–cones $\hat{s}_r \in \hat{\Psi}$:

$$s_l^{j_l} = \underset{r=1\ldots t}{\cup} \hat{s}_r \qquad (5.4)$$

and we get $\hat{s} \cap s_l^{j_l} \underset{\text{Eq.(5.4)}}{=} \hat{s} \cap \left(\underset{r=1\ldots t}{\cup} \hat{s}_r\right) = \underset{r=1\ldots t}{\cup} (\hat{s} \cap \hat{s}_r) \underset{\text{Eq.(5.3)}}{\neq} \emptyset$. Then an index r^* exists with $\hat{s} = \hat{s}_{r^*}$, i.e.

$$s_l^{j_l} = \hat{s}_1 \cup \ldots \cup \hat{s} \cup \ldots \cup \hat{s}_t \quad . \qquad (5.5)$$

On the other hand, from (5.3) it follows that one can find an element $\tilde{u} \in \hat{s} \cap s_l^{h_l}$ which implies $\tilde{u} \in \hat{s}$ and therefore with (5.5) $\tilde{u} \in s_l^{j_l}$. Yet, this is a contradiction to (5.2) and the lemma is shown.□

References

[1] K. Hering, R. Haupt, and T. Villmann. An Improved Mixture of Experts Approach for Model Partitioning in VLSI–Design Using Genetic Algorithms. Technical Report 14, University of Leipzig / Inst. of Informatics, Germany, 1995.

[2] M. I. Jordan and R. A. Jacobs. Hierarchical Mixture of Experts and the EM Algorithm. In P. Morasso, editor, *Proc. ICANN'94*, pages 479–486. Springer, 1994.

[3] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49(2):291–307, 1970.

[4] J. R. Koza. *Genetic Programming*. MIT Press, 1992.

[5] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner-Verlag Stuttgart and JOHN WILEY & SONS, 1990.

[6] N. Manjikian. High performance parallel logic simulation on a network of workstations. TechReport T-220, CCNG, University of Waterloo, 1992.

[7] R. B. Mueller-Thuns, D. G. Saab, R. F. Damiano, and J. A. Abraham. VLSI Logic and Fault Simulation on General Purpose Parallel Computers. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12:446–460, 1993.

[8] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithm in Combinatorial Optimization. *Parallel Computing*, (7):65–88, 1988.

[9] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Information*. Fromman Verlag Freiburg (Germany), 1973.

[10] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley, Reading, MA, 1992.

[11] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley and Sons, 1981.

[12] S. P. Smith, B. Underwood, and M. R. Mercer. An analysis of several approaches to circuit partitioning for parallel logic simulation. In *Proceedings IEEE International Conference on Computer Design (ICCD)*, pages 664–667, 1987.

[13] C. Sporrer. *Verfahren zur Schaltungspartitionierung für die parallele Logiksimulation*. Verlag Shaker Aachen, 1995.

[14] W. G. Spruth. *The Design of a Microprocessor*. Springer Berlin, Heidelberg, 1989.